Mobile Computing

Abstract

Advances in wireless networking have prompted a new concept of computing, called mobile computing in which users carrying portable devices have access to a shared infrastructure, independent of their physical location. This provides flexible communication between people and (ideally) continuous access to networked services. Mobile computing is revolutionizing the way computers are used and in the coming years this will become even more perceptible although many of the devices themselves will become smaller or even invisible (such as sensors) to users.

This essay attempts to give an insight into mobile computing, in particular, software design issues (models, algorithms and in particular middleware) are considered. Such issues arise from the need for wireless networking, the ability to change location and the need for unencumbered portability as well insuring security standards comparable to that found is distributed systems or central systems.

1 Introduction

In the last 10 years, the advent of mobile phones as well as laptops has dramatically increased the availability of mobile devices to businesses and home users. More recently, smaller portable devices such as PDAs and especially embedded devices (e.g. washing machines, sensors) have slowly changed the way humans live and think of computers.

Computing is drifting away from just being concentrated on computers and relates more and more towards society, its people and its infrastructures. This is particular true where sensors are being developed to be so minute that they are literally embedded in clothing and even humans![1]

Mobile computing is associated with the mobility of hardware, data and software in computer applications. The study of this new area of computing has prompted the need to rethink carefully about the way in which mobile network and systems are conceived. Even though mobile and traditional distributed systems may appear to be closely related, there are a number of factors that differentiate the two, especially in terms of type of device (fixed/mobile), network connection (permanent/intermittent) and execution context (static/dynamic)[2].

Section 2 describes the main characteristics and differences between mobile and fixed distributed systems. Section 3 attempts to give a brief insight into the theory of mobile

[1] http://www.kevinwarwick.com/

[2] Framework of the characterisation of mobile distributed systems from the paper *Mobile Computing Middleware* by Cecilia Mascolo et al.

computing, in particular models and algorithms. Section 4 describes the main features desired for a mobile system middleware and in particular LIME will be used as an example of such middleware systems, which is based on tuple spaces. Section 5 concludes the paper.

2 Different types of Mobile Systems

In many ways, mobile computing has several characteristics reminiscent of distributed systems. In order to understand mobile systems, one must first understand where the similarities and the differences of distributed and mobile systems lie. The following section is an explanation of the different types of distributed systems ranging from the traditional type to nomadic, ad-hoc and finally ubiquitous ones.

2.1 Traditional Distributed System

Traditional distributed systems consist of a collection of fixed hosts that are themselves attached to a network– if hosts are disconnected from the network this is considered to be abnormal whereas in a mobile system this is quite the norm. These hosts are *fixed* and are usually very powerful machines with fast processors and large amount of memory. The bandwidth in traditional systems is very high too.

Furthermore, the execution context is said to be *static* as opposed to a *dynamic* context whereby host join and leave the network frequently. In a traditional system, location rarely changes as well and hosts are much less likely to be added or deleted from the network.

Traditional distributed systems also need to guarantee non-functional requirements[3] such as scalability (accommodate a higher load at some time in the future), openness (possibility to extend and modify the system easily), heterogeneity (integration of components written using different programming languages, running on different operating systems, executing on different hardware platforms), fault-tolerance (recover from faults without halting the whole system) and finally resource-sharing (some form of access control).

2.2 Nomadic Distributed System

This kind of system is composed of a set of mobile devices and a core infrastructure with fixed and wired nodes. Mobile devices move from location to location, while maintaining a connection to the fixed network. There are problems that arise from such shifts in location. The mobile host has a home IP address and thus any packets sent to the mobile host will be delivered to the home network and not the foreign network where the mobile host is currently located. Such problem can be solved by forwarding packets to the foreign network with the help of Mobile IP. Nevertheless, Mobile IP also suffers from efficiency (routing issues), QoS, security (authentication of mobile host at foreign network and end-to-end security required) and wireless access (reduced capacity) problems.

[3] *Mobile Computing Middleware* by Cecilia Mascolo et al.

These systems are susceptible to the uncertainty of location, a repeated lack of connections and the migration into different physical and logical environments while operating. However, compared to ad-hoc networks, nomadic systems still have comparatively reliable connections and services since most of these are actually supported by the fixed infrastructure ("backbone") of the network.

The non-functional requirements mainly differ, compared to the traditional distributed systems, in the heterogeneity (affected by the presence of both fixed and mobile devices across the network as well as the variations in technologies (e.g.: wireless)), resource sharing (must take into account different issues when the resources need to be discovered) and fault tolerance of the system (considered to be quite the norm). Quality and provision of these resources must be carefully considered too.

2.3 Ad-Hoc Mobile Distributed System

Ad-hoc distributed systems are possibly the only type of network that comes close to mobile networks in the sense that every node is literally *mobile*. It is these networks that are very much seen as the systems of the future, whereby hosts are connected to the network through high-variable quality links (e.g.: from GPS to broadband connection) and executed in an extremely dynamic environment.

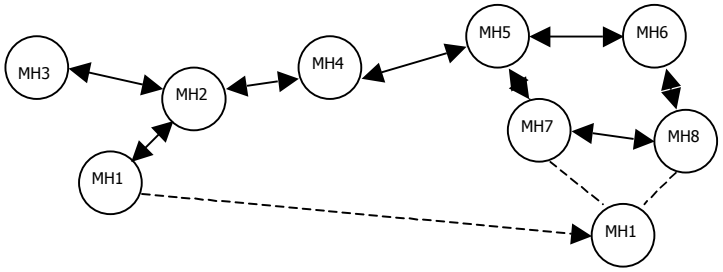
A-hoc systems do not have any fixed infrastructure which differs them both from traditional and nomadic distributed systems. In fact, ad-hoc networks may come together as needed, not necessarily with any assistance from the existing (e.g.: Internet) infrastructure. When nodes are detached from the fixed/mobile network they may evolve independently and groups of hosts opportunistically form "clusters" of mini-networks. The speed and ease of deployment make ad-hoc networks highly desirable.

These kinds of systems are extremely useful in conditions where the infrastructure is absent, impractical to establish or even expensive to build (e.g.: military applications, high terrain uses, and emergency relief operations).

However, being a relatively new technology, this field of networking demands a lot of research to be done to improve it, especially its non-functional requirements as well as algorithms for routing protocols (e.g.: distance vector and dynamic source routing algorithms).

Security threats have to be dealt even more cautiously in ad-hoc networks. Designing secure key distribution in an ad-hoc network might be an extremely hard task. Any reliance on a certificate of authority is not trivial at all, for the same reasons that reliance in any centralized authority is problematic. Additional problems include the increased packet sizes required by authentication extensions, unicast/multicast routing, Quality of Service support and power aware routing. Furthermore, due to the limited transmission range of wireless network interfaces, multiple *hops* may be needed to exchange data between nodes in the network (c.f. MANET).

Ad Hoc network of mobile nodes – these are able to move relative to each other. MH1 moves away from MH2 and establishes new links with MH7 and MH8. Most algorithms also allow for the appearance of new mobile nodes and the disappearance of previously available nodes.



3 Theory in Mobile Computing

This section is only an introductory section so it will not go into any detail other than state what the current trend of search in these two fields related to mobile computing are.

3.1 Models

Models permit the precise description of existing languages and system semantics. In fact, they enable the formal reasoning about the correctness of such semantics. Models are very much used to emphasize parallels and distinctions among various forms of mobility (logical and physical) and are concerned with the formulation of appropriate abstractions useful in specification and evaluation of such mobile systems.

Models are mainly concerned with the characteristics of mobile units such as the unit of mobility (who is allowed to move), its location (where a mobile unit is positioned in space) and its context (determined by the current location of mobile units). There are many existing models and many more are still in research[4]:

- Random mobility model(s)
- Markovian model
- Exponential Correlated Random Model
- Nomadic Community Model

3.2 Algorithms

The current algorithms applied reflect the assumptions that are made about the underlying system. Unfortunately, many of these assumptions are not suited for current algorithms for mobile systems. Mobile algorithms are obliged to treat in much detail space and coordination of mobile systems. In particular, algorithms have to carefully take into consideration location changes, the frequency of disconnection, power limitations and the dynamic changes in the connectivity pattern of mobile systems. This field of theory is in fact spread among a vast spectrum of research due to the large diversity of mobile systems[5].

[4] students.cec.wustl.edu/~cs673/mobilityModels.ppt
[5] http://www.cs.wustl.edu/mobilab/research.html

4 Middleware

A lot of research has been made in recent years into the translation of traditional middleware into that of mobile distributed systems. However, this is not as easy as researchers first thought due to the differences between traditional and mobile systems stated previously. Traditionally, middleware for physical mobility[6] has been application centred (e.g.: Bayou system). Nevertheless, this approach is not suitable for a generic form of mobile computing and therefore a general purpose middleware becomes a necessity. Hiding mobility is increasingly more difficult or even meaningful, thus a new core of abstractions that extend distributed middleware with support to mobility must be fully researched.

Another reason for the so-called "lack" of a suitable mobile system middleware is that traditional distributed systems have been around for over 20 years whereas mobile systems are a new technology which has been around at most 10 years. This means that this new field is still very much into research.

Middleware for nomadic and ad-hoc mobile distributed systems has a set of comparable characteristics that influence how the middleware should in fact behave. Mobile devices require light computational load – existing middleware for heavy computational load such as that found in traditional distributed systems cannot be applied. The intermittent connection nature of mobile systems also requires an asynchronous form of communication. Additionally, unlike fixed distributed systems, mobile systems execute in an extremely dynamic context which in turn necessitates that devices be aware at all time of their environment (e.g.: type of connection they are switching to).

Middleware for mobile distributed systems is split into different research areas such as context-aware middleware (*principle of Reflection*); location-aware middleware (e.g.: *Nexus*); data sharing-oriented middleware (e.g.: *Bayou, Coda, Odyssey*); tuple space-based middleware (e.g.: *Lime, TSpaces*). This paper will only have a brief look at the tuple space-based approach and, in particular, Lime will be used as an example.

4.1 Tuple Space Middleware

The classical definition of a tuple space is that of a shared associative memory consisting of a collection of tagged data records (*tuples*)[7]. Tuples may be created and placed in the tuple space and they can be access concurrently by several processes with blocking primitives. In a mobile context a completely asynchronous (de-couple communication) and de-coupled model is suitable and thus tuple space middleware is appropriate. Tuple spaces needn't depend either on the machine or platform in which they are running.

Tuple spaces have proven to be suitable for mobile computing because of the dynamic context nature (migration and connectivity patterns) of mobile systems. They are an attractive approach for coordinating mobile units across a mobile computing environment.

[6] i.e. movement of mobile hosts in a building or even large regions of the planet
[7] *Concurrency: State models and Java Programs*, J.Magee & J.Kramer

4.2 LIME

Lime (**L**inda **I**n a **M**obile **E**nvironment) is a java-based middleware that provides a coordination layer that can be exploited successfully for designing applications that exhibit either logical or physical mobility - or both. The fundamental design criteria underlying Lime comes from the realization that the defining problem of mobile computing is dealing with, and exploiting, a dynamically changing context. To achieve its goal, Lime borrows and adapts the communication model made popular by Linda[8].

Lime provides coordination among processes (*Lime agents)* via a shared memory mechanism. When processes use Lime to coordinate, no messages are explicitly sent to other processes. All communication occurs through access to a shared medium, namely the *tuple space*.

The difference between Linda and Lime is that the former breaks up the tuple space of the latter into several tuples spaces. Each one of these tuple spaces is associated to a mobile unit[9] and every mobile unit has access to an *interface tuple space* that is constantly and exclusively attached to that unit and transferred along with it when movement occurs. The tuple space that can be accessed through this interface is thus shared by construction and is temporary because its content changes according to the movement of mobile units.

Lime is a very powerful tuple space-based middleware since it promotes the reduction of details of mobility (*transparency*) and the distribution to changes to what is perceived as the local tuple space. This allows designers of mobile systems to be alleviated from the burden of constantly maintaining a view of the context consistent with changes in the configuration of the system. However, transparency could lead to an oversimplification of a system. In some cases, where there is a need to achieve a high amount of context awareness, this may not be favorable.

Nonetheless, Lime offers several "solutions" such as making information about the system configuration available via LimeSystem[10]; allow actions to be taken in response to a change in the configuration of the system by setting reactions on the tuple space; and broaden Linda operations with tuple location parameters that permit to function on different projections of the transiently collective tuple space.

5 Conclusion

There is little doubt that mobile computing will enhance many aspects of the lives of humans. One must wonder though whether or not everyone will want to have such an "invading" technology, especially when it comes to ubiquitous computing. Some people may be

[8] Coordination model designed in the mid 1980's at Yale University which enables the development of distributed, shared memory applications.
[9] Either a mobile agent or a mobile host.
[10] Read-only transiently shared tuple space which contains details about the mobile components present in the community and their relationship.

scared with regards to health issues (electrical component in clothes/skin may have unwanted long term effects on humans) as well as privacy ones (a chip inserted into someone's arm could be made as a tracking device without the wearer knowing; "Warchalking" – drawing chalk symbols on a wall/pavement indicating the presence of a wireless networking node, especially in business areas. This has raised the issue of security in wireless networks and has allowed people to freely enjoy "surfing").

However, these worries are more "ethical" and "social" than actual technological. There is little doubt that mobile computing offers a potential large economic market in networks. Nevertheless, the greatest challenges that are to be solved are namely security, portability, and scalability and power control issues.

This promising field of research is still at an early stage but advances are made to improve the quality and the availability of mobile systems. Consumers will be affected more and more by the emancipation in the availability of mobile devices thanks to a reduction of price in manufacturing as well as a reduction in the size of such devices (ubiquitous computing).